APPLICATION UNDER UNITED STATES PATENT LAWS

Atty. Dkt. No. PW 279174
(M#)

Invention: CRYPTOGRAPHY-BASED TAMPER-RESISTANT SOFTWARE DESIGN MECHANISM

Inventor (s): Michael A. NENASHEV

Pillsbury Winthrop LLP
Intellectual Property Group
1600 Tysons Boulevard

McLean, VA 22102
Attorneys
Telephone: (703) 905-2000

This is a:

- ☐ Provisional Application
- ☒ Regular Utility Application
- ☐ Continuing Application
  - ☐ The contents of the parent are incorporated by reference
- ☐ PCT National Phase Application
- ☐ Design Application
- ☐ Reissue Application
- ☐ Plant Application
- ☐ Substitute Specification
  Sub. Spec Filed _____
  in App. No. _____ / _____
- ☐ Marked up Specification re
  Sub. Spec. filed _____
  In App. No _____ / _____

SPECIFICATION

Intel Ref: P11051
Pillsbury Ref: 81674/279174

CRYPTOGRAPHY-BASED TAMPER-RESISTANT SOFTWARE DESIGN MECHANISM

BACKGROUND

[0001] Aspects of the present invention relate to software. Other aspects of the present invention relate to software security.

[0002] Tampering with software involves unauthorized access and modification to software. Such acts often directly associate with security issues. For example, altering network security software to perform what it is not designed to do may pose a serious threat to network security. Similarly, changing application software that transfers secure data from one computer system so as to expose that secure data may compromise that secure data.

[0003] To ensure software integrity, different protection mechanisms have been attempted. The most common practice to protect data access is the use of passwords. With a password based mechanism, an operator who initiates the data (software) access supplies a password which is then authenticated against a matching pre-determined password that is either hard coded in the software or stored in, for example, a file on the filesystem. With a password mechanism, it is assumed that both the operator and the software are trusted parties during data manipulation.

[0004] Another approach to secure software access is through access right control. For example, secure software may only be accessed with a certain level of access right such as administrator's privilege. Many Unix systems allow designated software to be executed at a higher level of permission than the default level of permission granted to the current login. Other types of security systems rely on a certificate authority. Such systems implement security measures by allowing a file system to "fingerprint" software at the system administrator's level.

Some advanced security systems enforce secure software access based on encryption key management mechanisms.

[0005] Conventional approaches to ensuring software and data integrity often depend on the underlying operating system implementation or other hardware components and sometimes require significant installation and maintenance effort. Static encryption key management mechanisms of some conventional approaches to ensure software and data integrity provide static, instead of dynamic, key management, which is inflexible and easier to compromise. Furthermore, the conventional approaches do not provide the means to identify software tampering that has been committed.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] The inventions presented herein are described in terms of specific exemplary embodiments, which will be described in detail with reference to the drawings. These embodiments are non-limiting exemplary embodiments, in which like reference numerals represent similar parts throughout the several views of the drawings, and wherein:

[0007] Fig. 1 depicts a high level architecture of an embodiment of the present invention;

[0008] Fig. 2 is an exemplary flowchart of a process, in which an unintended execution of a software program is prevented based on a fingerprint computed dynamically from a protected portion of the program, according to an embodiment of the present invention;

[0009] Fig. 3 depicts a high-security program protection mechanism of an embodiment of the present invention;

[0010] Fig. 4 is an exemplary flowchart of a process, in which high-security is initially set up to protect a secure portion of a program, according to an embodiment of the present invention;

[0011] Fig. 5 is an exemplary flowchart of a process, in which high-security protection, set up for a secure program, is enforced, according to an embodiment of the present invention;

[0012] Fig. 6 depicts a high level functional diagram of a secure document replication mechanism according to an embodiment of the present invention; and

[0013] Fig. 7 is an exemplary flowchart for a secure document replication mechanism according to an embodiment of the present invention.

DETAILED DESCRIPTION

[0014] The invention is described below, with reference to detailed illustrative embodiments. It will be apparent that the invention can be embodied in a wide variety of forms, some of which may be quite different from those of the disclosed embodiments. Consequently, the specific structural and functional details disclosed herein are merely representative and do not limit the scope of the invention.

[0015] Fig. 1 depicts a high level architecture of a tamper-resistant high-security software protection mechanism 100 and the environment in which it operates. The tamper-resistant high-security software protection mechanism 100 sets up and enforces cryptography-based high-security protection of an informational resource based on a protected portion 115 of a program 110 stored in a segment of a memory 105. As will be apparent to those skilled in the art, a protected portion 115 of the program 110 may compose the whole program 110. In Fig. 1, the tamper-resistant high-security software protection mechanism 100 includes a high-security set-

up mechanism 130 and a high-security protection mechanism 140, both of which operate with the protected portion 115 of the program 110 to achieve tamper-resistant high-security protection. Optionally, a low-security protection mechanism 150, as shown in Fig. 1, may provide low-security protection to the program 110. A data processing mechanism 160 accesses the protected portion 115 through a secure channel 170 authorized by the tamper-resistant high-security software protection mechanism 100 only when no software tampering took place with respect to the protected portion 115.

[0016] The tamper-resistant high-security software protection mechanism 100 provides security measures against an act to gain access to protected data through program tampering. A program may include source code that implements objects and methods as well as the corresponding executable code, compiled from the source code and installed and running in memory 105. A program may access secure information that needs to be protected. Such secure information may include files on a file system, attributes from a table in a database, or any other electronic data from an informational data resource that can not be compromised.

[0017] A tampering act may include changing source code of a program, (hence also the corresponding executable code after the compilation) to expose the values of protected variables. When any tampering act is detected, the tamper-resistant high-security software protection mechanism 100 prevents the protected portion of the secure program from being executed.

[0018] The tamper-resistant high-security software protection mechanism 100 traces software tampering through a dynamically computed fingerprint (120a or 120b). A fingerprint is a unique identifier of the protected portion 115 and is computed on-the-fly based on some invariant characteristics of the protected portion 115. Invariant characteristics used to generate a fingerprint may be determined in such a way that they are invariant with respect to different

executions, if there is no tampering act, yet sensitive to any change introduced by tampering acts. For example, the content of a random access memory (RAM) block allocated to an object does not change between executions unless the source code is changed. In this case, the region between the starting and the ending address of such a block may be used as an invariant characteristic in computing a fingerprint. As will be apparent to those skilled in the art, any number of invariant characteristics can be used to compute a fingerprint and any number of algorithms can be used to generate the fingerprint based on invariant characteristics.

[0019] The invariance may be defined with respect to certain scope. For example, memory allocation strategy may differ from computer to computer. In this case, the invariance may be defined within the scope of a physical machine. A program may have different versions and each may correspond to a different fingerprint due to different invariant characteristics. Furthermore, the selection of invariant characteristics may depend on the nature of the program and the application environment in which the program operates.

[0020] The fingerprint (120a and 120b) of the protected portion 115 is computed based on invariant characteristics. As long as the protected portion 115 is not tampered, the message digest computed from the non-tampered protected portion remains identical. The fingerprint 120a is generated during the initial set-up and is immediately used to encrypt the supplied authentication information. The encrypted authentication information is then stored for future use of detecting tampering on the program 110.

[0021] The authentication information may include an aministrator-level account/password pair or a token. To initially set up tamper-resistant high-security protection for the protected portion 115 of the program 110, an administrator with high-security authority activates the high-security set-up mechanism 130 and provides the authentication information.

[0022] In the tamper-resistant high-security software protection mechanism 100, a symmetrical encryption scheme may be applied to the high-security authorization information 135 to set up tamper-resistant protection to the data accessed by the program 110 from one or more informational resources. With a symmetrical encryption scheme, the high-security authorization information 135 is encrypted, with a dynamically generated key, during the initial set-up process. Once the encrypted high-security authorization information is saved, it can be retrieved and decrypted only when an exactly identical key is used to decrypt it. Utilizing this property of a symmetrical encryption, the tamper-resistant high-security software protection mechanism 100 uses a fingerprint that is tightly coupled with the invariant characteristics of a secure program or the protected portion 115 , as a key for both encryption and decryption purposes.

[0023] In Fig. 1, the fingerprint 120a, computed during a high-security protection set-up process, is used, by the high-security set-up mechanism 130, as the key to encrypt the high-security authorization information (supplied by the administrator). The encryption may be performed once to generate an encrypted high-security authorization information 145 which is later decrypted in a symmetrical encryption scheme, in an attempt to detect changes made to the protected portion 115, which may constitute an attempt to gain access to data from protected resources.

[0024] The tamper-resistant high-security protection may need to be set-up again whenever there are operating environment changes. For example, when a new compiler is installed, the fingerprint 120a, as the encryption key in a symmetrical encryption scheme, may have to be re-computed if runtime memory allocation addresses are used in computing the fingerprint 120a. When the computer on which the program 110 is installed is upgraded (e.g.,

memory size is increased), the fingerprint 120a may also need to be re-computed. Yet another different factor that affects the fingerprint 120a is when a different version of the program is installed. If the new version of the program 110 includes changes to the original source code, the corresponding new fingerprint may substantially differ from the fingerprint computed from the previous version of the program.

[0025] The fingerprint 120a encodes the normal state of the protected portion 115 of the program 110. When it is used to encrypt the high-security authorization information, the symmetrical encryption scheme ensures that only a key that encodes an identical normal state of the program 110 can be successfully used to decrypt the encrypted high-security authorization information. That is, if there is any change in a fingerprint computed, at runtime, from the protected portion 115 of the program 110, such change indicates that the protected portion 115 at the runtime is not identical to the original version. Hence, software tampering may have occurred. In this case, the authentication information may not be retrieved and access to information can not succeed. Such runtime protection is enforced through the high-security protection mechanism 140.

[0026] During runtime, when the protected portion 115 is accessed, the high-security protection mechanism 140 is activated. To enforce the high-security protection (which detects any tampering on the protected portion 115), the encrypted high-security authorization information is retrieved based on the fingerprint 120b generated on-the-fly using the characteristics of the runtime protected portion 115. The high-security protection mechanism 140 uses the fingerprint 120b, as a key to decrypt the retrieved encrypted high-security authorization information 145 that is set up during the initial installation process performed by, for example, an administrator. If the protected portion 115 is not tampered, the fingerprint 120b is identical to

the fingerprint 120a. In that case, the fingerprint 120b can lead to a successful decryption. Otherwise, the decryption attempt will fail, thus indicating that the protected portion 115 of the program 110 has been tampered with.

[0027] There are different approaches to computing a fingerprint. For example, a hash function may be used to compute a one-way hash on the portion of the RAM where the protected portion 115 resides. Such a hash function may traverse the byte array of the protected portion 115 and generate a message digest of the protected portion as the fingerprint 120a or 120b. There are existing standard methods such as the Secure Hash Algorithm (SHA) (SHA-1 is a method for secure hashing; it is approved by the U.S. Federal Information Processing Standards (FIPS) and specified in FIPS 180-1, Secure Hash Standard (SHS), currently located at URL:http://www.itl.nist.gov/fipspubs/fip180-1.htm) that generate a message digest based on a hash function. A message digest generated by a SHA possesses important properties. For example, it is computationally infeasible to hash two different input byte arrays to a same digest. In addition, a message digest computed using a hash function does not reveal anything about the input used by the hash function.

[0028] At runtime, multi-layer protection may also be optionally supported. For example, to activate the program 110, a data processing mechanism 160 (which may correspond to a person or an application) may need to supply a low-security authorization information, such as a password to the low-security protection mechanism 150. Only when the low-security protection mechanism 150 grants the access based on, for example, password clearance, may the program 110 be executed. Activation of the high-security protection may be transparent to the data processing mechanism 160. The high-security protection may be automatically triggered whenever the protected portion 115 is accessed.

[0029] Fig. 2 is an exemplary flowchart of a process, in which a protected portion (e.g., 115) of a program (e.g., 110) is used to prevent unauthorized access to protected resource according to an embodiment of the present invention. In Fig. 2, tamper-resistant high-security protection is first initialized. The high-security authorization information, provided by an authoritative administrator, is first encrypted, at 210, using a fingerprint (e.g., 120a), computed from the protected portion during the set-up process, as the encryption key.

[0030] At runtime, whenever the protected portion 115 is accessed at 220, the encrypted high-security authorization information is retrieved and is decrypted, at 230, using a fingerprint (e.g., 120b), computed from the protected portion of the program at runtime, as the key. If the decryption is successful (i.e., the protected portion 115 is not tampered), a secure channel is authorized at 240.

[0031] Fig. 3 depicts a detailed high level functional block diagram of the tamper-resistant high-security software protection mechanism 100 of an embodiment of the present invention. In Fig. 3, the protected portion 115 within the program 110 includes an encryption function 310 and a set of objects and methods 320. The encryption function 310 is responsible for computing a fingerprint 120 based on some invariant characteristics of the protected portion 115. The encryption function 310 may be realized as a hash function. For example, a SHA hash function may be adopted. The encryption function 310 resides inside of the protected portion 115 and is triggered to compute a fingerprint whenever the protected portion is accessed.

[0032] In Fig. 3, the high-security set-up mechanism 130 comprises an encryption mechanism 330, which encrypts, using the fingerprint 120 as a key, the high-security authorization information 135 to generate encrypted authorization information 145, and an encrypted authorization information storage 340, which stores and provides the encrypted

authorization information 145. The stored encrypted authorization information 145 is later used at runtime to enforce high-security protection to data from secure information resources.

[0033] The high-security protection mechanism 140 comprises a high-security information access mechanism 350, which retrieves the encrypted authorization information 145 from the storage 340, and a decryption mechanism 360 that decrypts the retrieved encrypted authorization information 145 using a message digest, as a key, computed from the protected portion 115 on-the-fly. The high-security information access mechanism 350 may be triggered when the protected portion is accessed. The decryption mechanism 360 is the counterpart of the encryption mechanism 330 in a symmetrical encryption scheme.

[0034] In Fig. 3, the exemplary data processing mechanism 160 includes a data access mechanism 370 and a data storage 380. The data access mechanism 370 initiates the access to the program 110. Such initiative may be from a human operator or from an application program, such as a regularly scheduled replication job. As shown in Fig. 3, the data access mechanism 370 may interface with a low-security protection mechanism 150 to initiate the access. For example, if the data access mechanism 160 corresponds to a human operator, low-security authorization information 175, such as a password or a token, may be provided.

[0035] The data storage 380 in the data processing mechanism 160 may be used to store information that may be accessible to the data access mechanism 370. Such information may be generated (written) by a method in the protected portion 115. For example, a secure data replication method in the protected portion 115 may duplicate data in a data storage for replication purposes (a secure data replication mechanism is discussed later in referring to Fig. 6 and Fig. 7). A protected method may also retrieve data from the data storage 380. For example, a secure data replication method may copy data from the data storage to another data storage. For

security reasons, any data exchange between the data storage 380 and the protected portion 115 is through a secure channel 170 which may be granted only when it is verified that the protected portion 115 is not tampered with.

[0036] Fig. 4 is an exemplary flowchart of a process, in which the high-security set-up mechanism 130 sets up tamper-resistant high-security protection using a protected portion (e.g., 115) of a program (e.g., 110), according to an embodiment of the present invention. The initial set-up process may be initiated by a a system administrator. A system administrator may install the program 110 and set up tamper-resistant high-security protection, prior to its use, to ensure that the data is accessed in a secure fashion.

[0037] In Fig. 4, the program 110 is first activated at 410. The program 110 includes a protected portion 115 that aggregates different items in the program 110 to provide protection to the accessed data by preventing the execution of the program 110 whenever any change in the protected portion 115 that may be due to an tampering act is detected. High-security authorization information 135 is received at 420. The high-security authorization information 135 is provided by, for example, the personnel who initiates the set-up process. To encrypt the high-security authorization information, the encryption function 310 located in the protected portion 115 computes the message digest 120a at 430, based on the invariant characteristics of the protected portion 115. The fingerprint 120a is used as an encryption key to encrypt, at 440, the high-security authorization information. The encrypted high-security authorization information 145 is then stored, at 450, in the encrypted authorization information storage.

[0038] Fig. 5 is an exemplary flowchart of a process, in which the high-security protection mechanism 140 enforces tamper-resistant high-security protection on a protected portion (e.g., 115) of a program (e.g., 110), according to an embodiment of the present invention.

To activate the program 110, low-security protection may be first enforced. This may involve authenticating the party (either a person or an application program) that initiates the activation of the program 110. In Fig. 5, low-security authorization information 175 is first received, at 510, from the initiating party. The low-security authentication is performed at 515 based on the received low-security authorization information 175. If the authentication fails, determined at 520, the initiation of the program is aborted at 525.

[0039] If the low-security authentication is successful, the program 110 is activated at 530. The unprotected portion 112 of the program 110 is not involved in ensuring high-security protection. When the protected portion 115 of the program 110 is accessed at 535, the tamper-resistant high-security protection is activated. The encrypted high-security authorization information 145, generated by the high-security set-up mechanism during the initial set up, is first retrieved, at 540, from the encrypted authorization information storage 340.

[0040] To decrypt the encrypted high-security authorization information 145, the encryption function 310 located in the protected portion 115 of the program 110 computes, at 550, the message digest 120b at runtime from the protected portion 115. The message digest 120b is used to decrypt, at 560, the encrypted high-security authorization information 145. If the protected portion 115 is tampered, the message digest 120b differs from the message digest 120a that is initially used to encrypt the high-security authorization information. In this case, the message digest 120b generated from the runtime protected portion 115 can not be successfully used as a decryption key to decrypt the high-security authorization information. The access to the protected data source is, therefore, denied at 580. If the decryption is successful, determined at 570, the access to the protected portion 115 is granted and a secure channel is authorized, at 590, between the protected portion 115 of the program 110 and the data storage 380.

[0041] Fig. 6 depicts a high level functional diagram of a secure data replication mechanism according to an embodiment of the present invention. In Fig. 6, secure data replication mechanism 600 utilizes the tamper-resistant high-security software protection mechanism 100 to realize secure data replication. In Fig. 6, a program 110 includes a replication method 610 that is aggregated into a protected portion 115 with other protected objects and/or methods (320) and encryption function 310. The function that the replication method 610 performs is to make a copy of the data stored in a source database, 620, and to duplicate the data in a destination database, 630.

[0042] To ensure safe data replication, the replication method 610 may be designed so that it does not reveal the content of the data that is being replicated. The security of the replicated data relies on such a design feature. If such design feature is compromised (e.g., by software tampering), the confidentiality of the secure data is also compromised. In Fig. 6, the tamper-resistant high-security software protection mechanism 100 is applied to protect the replication method 610 against tampering. With the tamper-resistant protection, if software tampering (e.g., the code of the replication method is changed to reveal the replicated data) is identified (by the tampering-resistant high-security protection mechanism), the data replication mechanism depicted in Fig. 6 will not be able to authorize access to the data sources (so that no data will be exposed).

[0043] Data replication may include data copying (from a source database, e.g., 620) and data duplicating (to a destination database, e.g., 630). In the exemplary embodiment illustrated in Fig. 6, both copying and duplicating may be protected against tampering. To establish a session to copy secure data from the source database 620, the protected replication method 610 is activated. Prior to copying, the tamper-resistant high-security protection is enforced. The

encryption function 310 computes a fingerprint 120b. The decryption mechanism 360 retrieves the stored encrypted authorization information and decrypts it using the fingerprint 120b as a key. If the decryption is successful (which means that the protected replication method is not tampered), the access channel is open and the replication method 610 proceeds with copying of the data stored in the source database 620.

[0044] Prior to establishing a session to duplicate the secure data in the destination database 630, the tamper-resistant high-security protection is enforced. The encryption function 310 computes the fingerprint 120b. If the fingerprint 120b enables a successful decryption of the encrypted high-security authorization information, the protected replication method proceeds with duplicating the secure data in the destination database 630.

[0045] Fig. 7 is an exemplary flowchart for a secure document replication mechanism 600 according to an embodiment of the present invention. The protected replication method is first activated, at 710, to copy data from a source database (620). Prior to the execution, encrypted high-security authorization information 145 is retrieved, at 715, from the encrypted authorization information storage 340. The encryption function 310 computes, at 720, the message digest 120b from the protected portion 115.

[0046] The generated message digest 120b is used, at 725, to decrypt the high-security authorization information 145. If decryption is not successful, determined at 735, the data replication is aborted. If decryption is successful, the protected replication method 610 copies data, at 730, from the source database (620).

[0047] Prior to duplicating data in the destination database (630), the encrypted authorization information is retrieved at 760. The encryption function 310 computes, at 770, the message digest 120b which is then used to decrypt the retrieved encrypted authorization

information at 780. If the decryption is successful, determined at 785, the replication method proceeds to duplicate, at 790, the data copied from the source database (620) to the destination document database (630). If decryption is not successful, the replication is aborted at 740.

[0048] The detailed descriptions may have been presented in terms of program procedures executed on a computer or network of computers. These procedural descriptions and representations are the means used by those skilled in the art to most effectively convey the substance of their work to others skilled in the art. The embodiments of the invention may be implemented as apparent to those skilled in the art in hardware or software, or any combination thereof. The actual software code or hardware used to implement the present invention is not limiting of the present invention. Thus, the operation and behavior of the embodiments often will be described without specific reference to the actual software code or hardware components. The absence of such specific references is feasible because it is clearly understood that artisans of ordinary skill would be able to design software and hardware to implement the embodiments of the present invention based on the description herein with only a reasonable effort and without undue experimentation.

[0049] A procedure is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. These operations comprise physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, objects, attributes or the like. It should be noted, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

[0050] Further, the manipulations performed are often referred to in terms, such as adding or comparing, which are commonly associated with mental operations performed by a human operator. No such capability of a human operator is necessary, or desirable in most cases, in any of the operations of the present invention described herein; the operations are machine operations. Useful machines for performing the operations of the present invention include general purpose digital computers, special purpose computer or similar devices.

[0051] Each operation of the method may be executed on any general computer, such as a mainframe computer, personal computer or the like and pursuant to one or more, or a part of one or more, program modules or objects generated from any programming language, such as C++, Java, Fortran, etc. And still further, each operation, or a file, module, object or the like implementing each operation, may be executed by special purpose hardware or a circuit module designed for that purpose. For example, the invention may be implemented as a firmware program loaded into non-volatile storage or a software program loaded from or into a data storage medium as machine-readable code, such code being instructions executable by an array of logic elements such as a microprocessor or other digital signal processing unit. Any data handled in such processing or created as a result of such processing can be stored in any memory as is conventional in the art. By way of example, such data may be stored in a temporary memory, such as in the RAM of a given computer system or subsystem. In addition, or in the alternative, such data may be stored in longer-term storage devices, for example, magnetic disks, rewritable optical disks, and so on.

[0052] In the case of diagrams depicted herein, they are provided by way of example. There may be variations to these diagrams or the operations (or operations) described herein

without departing from the spirit of the invention. For instance, in certain cases, the operations may be performed in differing order, or operations may be added, deleted or modified.

[0053] An embodiment of the invention may be implemented as an article of manufacture comprising a computer usable medium having computer readable program code means therein for executing the method operations of the invention, a program storage device readable by a machine, tangibly embodying a program of instructions executable by a machine to perform the method operations of the invention, or a computer program product. Such an article of manufacture, program storage device or computer program product may include, but is not limited to, CD-ROM, CD-R, CD-RW, diskettes, tapes, hard drives, computer system memory (e.g. RAM or ROM), and/or the electronic, magnetic, optical, biological or other similar embodiment of the program (including, but not limited to, a carrier wave modulated, or otherwise manipulated, to convey instructions that can be read, demodulated/decoded and executed by a computer). Indeed, the article of manufacture, program storage device or computer program product may include any solid or fluid transmission medium, whether magnetic, biological, optical, or the like, for storing or transmitting signals readable by a machine for controlling the operation of a general or special purpose computer according to the method of the invention and/or to structure its components in accordance with a system of the invention.

[0054] An embodiment of the invention may also be implemented in a system. A system may comprise a computer that includes a processor and a memory device and optionally, a storage device, an output device such as a video display and/or an input device such as a keyboard or computer mouse. Moreover, a system may comprise an interconnected network of computers. Computers may equally be in stand-alone form (such as the traditional desktop personal computer) or integrated into another apparatus (such as a cellular telephone).

[0055] The system may be specially constructed for the required purposes to perform, for example, the method of the invention or it may comprise one or more general purpose computers as selectively activated or reconfigured by a computer program in accordance with the teachings herein stored in the computer(s). The system could also be implemented in whole or in part as a hard-wired circuit or as a circuit configuration fabricated into an application-specific integrated circuit. The invention presented herein is not inherently related to a particular computer system or other apparatus. The required structure for a variety of these systems will appear from the description given.

[0056] While this invention has been described in relation to preferred embodiments, it will be understood by those skilled in the art that other embodiments according to the generic principles disclosed herein, modifications to the disclosed embodiments and changes in the details of construction, arrangement of parts, compositions, processes, structures and materials selection all may be made without departing from the spirit and scope of the invention. Changes, including equivalent structures, acts, materials, etc., may be made, within the purview of the appended claims, without departing from the scope and spirit of the invention in its aspects. Thus, it should be understood that the above described embodiments have been provided by way of example rather than as a limitation of the invention and that the specification and drawing(s) are, accordingly, to be regarded in an illustrative rather than a restrictive sense. As such, the present invention is not intended to be limited to the embodiments shown above but rather is to be accorded the widest scope consistent with the principles and novel features disclosed in any fashion herein.